DATABASE ADMINISTRATION
ORACLE STANDARDS

Overview

Standards are established rules, principles, or measures that are widely used, available, or supplied, and recognized and accepted as having permanent value. These Oracle standards identify steps necessary to implement an Oracle application database at the Centers for Medicare and Medicaid Services (CMS). The standards are intended to compliment the methodology and procedures described in the *Roles and Responsibilities* document and Oracle reference manuals.

These standards must be followed with care and consideration given to database object naming conventions, appropriate database object usage, and required object parameter settings. Any requests for deviation from these standards must be submitted in writing to, reviewed by, and approved by the Central Oracle DBA staff at CMS.

Oracle Database Development Life Cycle

1.0 Development Phase

1. The Division of Data Services (DDS) is formally notified of the new project initiative and a Central Data Administrator (DA) and Central Database Administrator (DBA) are assigned. If DDS is going to provide Local Data or Database Administration support, these resources are assigned as well.

    - To initiate a DDS project, contact the DDS Director. You will be asked to submit a *Database Development Form*.
    - You should submit an initial project plan to DDS for review and approval of Data Administration and Database Administration tasks and schedule.

2. Requirements are gathered, analyzed, and documented by project or application analysts.

3. The Local Data Administrator determines the project data requirements and develops a preliminary logical data model. All models must be developed according to *DDMSS Data Administration standards*.

4. The logical model is reviewed and approved by the Central DA. It is recommended that the Local and Central DBA be involved in the review as well. For more information, see the *DDMSS Data Administration Standards.*

5. A preliminary physical model can now be developed by the Local DA/DBA and submitted to the Central DBA for review and conditional approval. At this point the Central DBA will create a new instance or modify an existing instance for the Local DBA to use in developing the database based on the approved model.

6. Local DBAs will be given access to create database objects within their own schemas, and should keep the Central DBA informed of all activity taking

place in the database. The Central DBA will provide all Oracle support related to the database software, space allocations, backup and recovery, and database troubleshooting.

- It is expected that through the normal development process, the data model will change to address issues related to application design, requirement changes, etc. The local DA and DBA can make these changes at their discretion, but should involve the Central DA/DBA in any significant changes to the model by scheduling an Application Architecture Review.
- The final version of the data model will be reviewed, and must be approved, by the Central DA/DBA staff before the database will be allowed to migrate to the test/validation server for testing and validation (which must be done prior to moving to production).

2.0 Test Validation Phase

1. When database development is complete, the database must be moved to the test/validation server for user testing, migration plan testing, and performance monitoring. However, before moving to the test/validation server, a Pre-Validation Migration Review shall be conducted.

2. Upon successful completion of the Pre-Validation Review, the Central DBAs will migrate the database to the test/validation server. Local DBAs will be given read-only access to this environment. User accounts for the application will be set up to support UAT or other testing. No changes to the database design can be made in this environment. All changes must be made in development and migrated to the test/validation server.

3. The local DBA, application developers, and system owners will manage the actual testing, and document the results and approvals.

3.0 Production Phase

1. Upon successful completion of testing on the test/validation server, the project team will scheduled a Pre-Production Migration Review with the Central DBAs in preparation for the move to production.

2. After approval by the Central DBA, the production move will be scheduled and performed by the DDS Central DBA staff. Local DBAs and developers will not be given access to the production environment.

3. Local DBAs should be available to the Central DBAs during the process to answer questions and provide assistance, if necessary.

4. Post production support from the local DBA should be available for a four week period following the implementation of the new or modified database in the production environment.

4.0 Maintenance Phase

1. The Central DBAs from DDS are responsible for all maintenance, backup and recovery, monitoring and tuning once the database is in production.

2. The Central DBAs, at their discretion, may make changes to the database to improve performance or stability. This would cover changes that would not require modifications to the applications that use the database. Changes requiring application modifications would be made through the normal application development process. All changes by Central DBA will be documented and maintained in the Erwin data model for that database.

5.0 Retirement of Development and Test Environments

The development and test environments for each project will remain in place for four weeks after production implementation. After that, the databases will be de-allocated from the development and test servers until they are needed for development purposes again. The development environment will be re-established upon reception of a Database Development Form, and the steps outlined above for the Oracle database development life cycle will be followed. The test environment will be recreated as part of the SDLC.

Oracle Database Design Standards

1.0 Oracle Design Overview

The Oracle Design Standards define the steps necessary to evolve a logical data model into a physical schema and ultimately a set of physical database structures in Oracle. At CMS, tools have been identified to facilitate this process. CA/Platinum ERwin should be used to translate the approved logical data model to a physical model.

While developing the physical database design, all standards must be followed. CMS Central Oracle DBAs are responsible for creating the instance, database, and tablespaces according to user requirements and available resources.

2.0 Instances

An Oracle instance refers to the System Global Area (SGA) and the database background processes. An instance is started (memory allocated and background processes started) and then a database (datafiles) is mounted by the instance. To start an instance, Oracle must read a parameter file. Parameters must be modified based on database requirements.

2.1 Instance Naming Standards

See Standard Naming Conventions.

2.2 Object Usage

STANDARD:

- Create the parameter file in the following directory: 'oracle/admin/dbname/pfile'
- Parameter file name must be: 'initdbname.ora'
- Control files must be suffixed with '.ctl'
- Instance name and DBNAME must be the same.

2.3 Required Parameters (DDL Syntax)

STANDARD: The parameters listed below must be modified in the init.ora parameter file defining an instance. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference library.

Parameter	Instructions
DBNAME	Provide a valid dbname. Refer to the Standard Naming Conventions. This name is written to the control file for the database.
INSTANCE_NAME	Specifies the name of the instance. CMS standards require the instance name to be the same as the database name.
DB_DOMAIN	Provide a valid domain name. This should be a valid network domain name. Domain name should be WORLD.
CONTROL_FILES	Specifies the name of the control files. Two control files must be specified on different mount points. A fully qualified directory path must be specified for the control files. Refer to the Standard Naming Conventions.
DB_BLOCK_SIZE integer	Specifies database block size. Recommended minimum block size is 4096 for OLTP databases. A larger block size is recommended for OLAP, DSS, and mixed databases. This value can not be changed once the database has been created.
DB_BLOCK_BUFFERS integer	Determines the number of buffers in the buffer cache in the SGA. This parameter may be modified as needed.
SHARED_POOL_SIZE	Specifies the size of the shared pool. The shared pool

integer	contains shared cursors, stored procedures, control structures, etc. Larger values improve performance in multi-user systems.
PROCESSES integer	Specifies the number of operating system user processes that can simultaneously connect to an Oracle server.
LOG_BUFFER integer	Specifies the amount of memory that is used when buffering redo log files. In general, larger values for the log buffer reduce I/O, particularly if the transactions are long and numerous.
LOG_CHECKPOINT_ INTERVAL integer	Specifies the frequency of checkpoints in terms of the number of redo log file blocks that are written between consecutive checkpoints.
LOG_CHECKPOINT_ TIMEOUT integer	Specifies maximum time in seconds before another checkpoint occurs. Setting the value to 0 disables time-based checkpoints and is not recommended. Default value for Enterprise Edition is 1800.
BACKGROUND_DUMP_ DEST	Specifies the pathname for a directory where trace files for background processes are written.
USER_DUMP_DEST	Specifies the pathname for a directory where user trace files are written.
LOG_ARCHIVE_START TRUE/FALSE	Indicates whether archiving should be automatic or manual at start up. TRUE indicates archiving is automatic, FALSE indicates that the database administrator will archive filled redo log files manually.
LOG_ARCHIVE_DEST_n	Specifies 1 through 5 destination parameters for archive logging. The pathname must be fully qualified for the archive log destination.
LOG_ARCHIVE_ FORMAT	Specifies the naming format of the archive log files. Recommended format is '%t_%s.dbf' where t = thread number, s = log sequence number.
CORE_DUMP_DEST	Specifies the pathname for a directory where core files are dumped.
AQ_TM_PROCESSES integer	Specifies whether a queue monitor is created. If set to 0 or not specified, then the queue monitor is not created. **Note:** during 8.1.6 migration this parameter was specified due to a problem during migration.
SORT_AREA_SIZE integer	Specifies the maximum amount of memory to use for sorts. Increasing size improves the efficiency of large sorts.
COMPATIBLE	Allows you to use new release while guaranteeing backward compatibility.

JOB_QUEUE_ PROCESSES integer	Specifies the number of SNPn background processes per instance. Job queue processes are used to process requests created by DBMS_JOB.
OS_AUTHENT_PREFIX	Authenticates users attempting to connect to the server with the user's operating system account name and password. The default value is OPS$.
ROLLBACK_SEGMENTS	Specifies one or more rollback segments to allocate to the instance. Never name the SYSTEM rollback segment.

2.4 Rollback Segments

Oracle recommends several goals of tuning Rollback Segments:

- Transactions should never wait for access to rollback segments.
- Rollback segments should not extend.
- No transaction should ever run out of rollback space.
- Readers should always be able to see the read-consistent images.
- Users and utilities should try to use less rollback.

Dynamic views that can be used for tuning:

- V$ROLLNAME: displays name and number of online rollback segments.
- V$ROLLSTAT: displays number of waits on the header transaction table, volume of bytes written by the transaction.
- V$SYSTEM_EVENT: the Undo Segment Tx Slot shows waits for the transaction slots and therefore contention on rollback segment headers.
- V$WAITSTAT: displays cumulative statistics of waits on header blocks. (The report.txt report from utlestat.sql indicates the time spent waiting on the header transaction tables.)
- V$SYSSTAT: displays the number of consistent and data block gets.
- V$TRANSACTION: displays current transactions using rollback segments.

Sizing rollback segments:

- Deletes need to store the before and after images in the rollback segment. Truncate is recommended before a load.
- Inserts store the ROWID in the rollback segment.
- Updates depend on how many columns are being updated.
- Indexed values generate more rollback, because the server process must change values in the index as well as in the table.

Rollback segment guidelines:

- Specifiy minimum extents 20.
- Specify OPTIMAL (minimum extents * extent_size).
- INITIAL and NEXT extent should be the same size.
- Specify initial storage parameter of 2K, 4K, 8K, 16K, etc. This will allow reuse of all freed space when extent is dropped.
- For infrequent large transactions (SQL/Loader programs) alter a large rollback segment online, alter small rollback segments offline, set MAXEXTENTS to UNLIMITED, alter MAXEXTENTS back after loads are complete.
- OLTP: One rollback segment for every 10 concurrent users.
- BATCH: One rollback segment for each concurrent job.
- Truncate table or partition before a delete.
- Use COMMIT and BUFFER when running an import.
- Set CONSISTENT=N for imports: this prevents the transaction from being set to read-only, and thus using too much segment space.
- SQL/Loader: for conventional path loading, set the COMMIT intervals with ROWS keyword.

Testing your size estimates:

1. Start with small rollback segments and force them to extend.
2. Create a rollback segment tablespace.
3. Select a number of rollback segments to test and create them in the tablespace.
4. Create the rollback segments so that all extents are the same size; choose an extent size that you suspect will need between 10 - 30 extents when the segments grow to full size.
5. Start with two extents.
6. Alter online the rollback segments you are testing.
7. Run the application.
8. Query against the V$ROLLSTAT and V$WAITSTAT views - V$ROLLSTAT gives info on shrinks, extends, wraps, etc., and V$WAITSTAT gives info on rollback contention.
9. Watch the maximum size a rollback segment extends.

3.0 Databases

A Database in Oracle is an object which is created to logically group other Oracle objects such as datafiles, tablespaces, tables, and indexes within the Oracle Data Dictionary.

3.1 Database Naming Standards

See Standard Naming Conventions.

3.2 Object Usage

STANDARD:

- Datafiles must be defined in a fully qualified path.
- Database name must follow the Oracle *Standard Naming Conventions*.
- Datafiles must end with '.log' for logs and '.dbf' for system files.

3.3 Required Parameters (DDL Syntax)

STANDARD: The parameters listed below must be included in the Oracle data definition language (DDL) when defining an application database. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference library.

Parameter	Instructions
DATABASE dbname	Provide a valid dbname. Refer to the Standard Naming Conventions. This name is written to the control file for the database.
CONTROLFILE REUSE	Normally used for re-creating an existing database. This will reuse an existing control file specified in the init.ora file. This option is not used when creating a new database.
DATAFILE dfname	Specifies one or more files to be used as system datafiles. Refer to the Standard Naming Conventions.
LOGFILE GROUP integer	Specifies one or more files to be used as redo logs. GROUP uniquely identifies a redo log file group. Values 1 to MAXLOGFILES. If this parameter is omitted then Oracle automatically generates its value. A database must have at least 2 redo log groups.
MAXLOGFILES integer	Specifies the maximum number of redo log file groups that can be created for the database.
MAXDATAFILES integer	Specifies the initial sizing of the datafiles section of the control file at CREATE DATABASE or CREATE CONTROLFILE time.
ARCHIVELOG	Specifies that the contents of a redo log group must be archived before the group can be reused. Required if point-in-time is a DB requirement. Default is NOARCHIVELOG.
CHARACTER SET	Specifies the character set the database uses to store data. The default value depends on the OS. For info on character sets see: *Oracle8i National Language Support*

	Guide.
NATIONAL CHARACTER SET	Specifies the national character set used to store data in columns specifically defined as NCHAR, NCLOB, or NVARCHAR2. The default value is dependent on CHARACTER SET. For more info on character sets see: *Oracle8i National Language Support Guide.*

4.0 Tablespaces

A tablespace is a logical structure where tables, indexes, and clusters are stored. A tablespace is an area of storage made up of one or more physical datafiles. One or more tablespaces make up the database.

There are two types of space management within tablespaces: dictionary-managed and locally managed tablespaces. In dictionary-managed tablespaces, the allocation of extents is tracked via Oracle's data dictionary.

4.1 Tablespace Naming Standards

See Standard Naming Conventions.

4.2 Object Usage

STANDARD:

- Assign one or more tables/indexes per tablespace.
- Create tablespaces explicitly using the CREATE TABLESPACE command rather than implicitly by creating a table without specifying a tablespace name.
- Create a least one tablespace as temporary for user sorts.
- Specify suffix '.dbf' for application datafile names, '.log' for log datafile names.
- Set the default storage parameters to account for the typical object size.
- Specify PCTINCREASE 0. Exceptions can only be made with Central DBA approval.
- Specify OPTIMAL for rollbacks segments.
- AUTOEXTEND must be off. This is the default.
- MAXEXTENTS must be specified.

4.3 Required Parameters (DDL Syntax)--Dictionary Managed

STANDARD: The parameters listed below must be included in the Oracle data definition language (DDL) when defining a dictionary managed application tablespace. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference library.

Parameter	Instructions
DATAFILE datafile path and name	Specify the mount point, directory, and name of the datafile. Datafiles created in $ORACLE_HOME/dbs directory will be removed without warning.
SIZE size parameter	Specifies the size of the file. Use K or M to specify kilobytes or megabytes.
DEFAULT STORAGE	The default storage parameters to account for the typical object size must be specified.
INITIAL size parameter	Specifies in bytes the size of the object's first extent. Use K or M. You cannot specify INITIAL in an ALTER statement. For rollback segments specify initial parameter in 8K, 16K, 32K, and 64K for small transactions. For large transactions, specify 128K, 256K, 512K, 1M, 2M, 4M.… See also Rollback Segments.
NEXT size parameter	Specifies in bytes the size of the object's next extent. Use K or M. For rollback segments use the same INITIAL and NEXT values.
MINEXTENTS	Specifies the total number of extents to allocate when the object is created. The default is 1 for application datafiles and 2 for rollback segments. Set rollback MINEXTENTS to 20.
MAXEXTENTS	Specifies the total number of extents, including the first, that Oracle can allocate for the object. The minimum is 1 for application datafiles and 2 for rollback segments. The default value depends on the OS block size. UNLIMITED is prohibited.
PCTINCREASE value	Specifies the percent by which the third and subsequent extents will grow over the preceding extent. Set PCTINCREASE to 0. This will prevent exponential growth during extent allocation.
OPTIMAL size parameter	For rollback segments only. Specifies an optimal size in bytes for the rollback segment. Oracle will dynamically deallocate extents when their data is no longer needed. Value cannot be less than the space initially allocated by MINEXTENTS, INITIAL, NEXT and PCTINCREASE.
TEMPORARY	Specifies that object will hold temporary objects, i.e., objects used by implicit sorts.

4.4 Required Parameters (DDL Syntax)--Locally Managed

STANDARD: The parameters listed below must be included in the Oracle data definition language (DDL) when defining locally managed tablespaces. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference library.

Parameter	Instructions
DATAFILE datafile path and name	Specify the mount point, directory, and name of the datafile. Datafiles created in $ORACLE_HOME/dbs directory will be removed without warning.
SIZE size parameter	Specifies the size of the file. Use K or M to specify kilobytes or megabytes. For rollback segments.
EXTENT MANAGEMENT LOCAL	Specifies that the tablespace is locally managed. AUTOALLOCATE: specifies that the tablespace is system managed, cannot specify the extent size. UNIFORM: specifies that the tablespace is managed with uniform extents. Specify size in K or M.

5.0 Tables

Oracle tables are the basic storage component in the Oracle database. Information is stored in tables in columns and rows. The data represented in these tables can be accessed using SQL data manipulation language (select, insert, update, and delete). Generally, tables are used in Oracle for applications and end-users to retrieve and manage business data.

5.1 Table Naming Conventions

See Standard Naming Conventions.

5.2 Object Usage

- Tables must be created in a tablespace that was explicitly created for the corresponding application. User tables may not be created in the system tablespace.
- There may be one or more tables per tablespace.
- All rows of a table should be uniquely identified by a column or set of columns to avoid duplicate rows. Every table defined in Oracle shall include an explicitly named primary key, and, if applicable, explicitly named foreign keys.
- All Oracle tables must have storage characteristics--see the section on Storage.

5.3 Required Components of the CREATE TABLE Statement

STANDARD: The components listed below must be included in the CREATE TABLE statement.

Component	Instructions
Schema	Must identify the application owning the given table. (If this parameter is omitted, the table is created in the user's own schema.)
Table Name	Must be a unique table identifier, following Standard Naming Conventions.

5.4 Optional Parameters in the CREATE TABLE Statement

The following parameters in the CREATE TABLE statement are optional:

- CONSTRAINT identifies the integrity constraint by the name constraint. There are two types of integrity constraints: column constraint and table constraint. Table constraint can impose rules in any column(s) of the table being created. A column constraint defines a rule for the current column being defined.
- NULL allows null values for a column.
- NOT NULL specifies that a column cannot contain null values--it must contain data or Oracle will not allow the row to be entered or updated.
- CHECK stipulates a condition that each row in the table must satisfy.
- UNIQUE designates a column or combination of columns as a unique key. You cannot define UNIQUE constraints on index-organized tables.
- PRIMARY KEY defines a column or combination of columns as the table's primary key.
- FOREIGN KEY defines a column or combination of columns as the foreign key in a referential integrity constraint.
- REFERENCES identifies the foreign key column that references the primary key of some other table. This establishes the relationship between two tables.
- ON DELETE CASCADE indicates that on deleting a column in the referenced table (the other table) corresponding rows in the table being created will be deleted.

5.5 Syntax

The general form of a CREATE TABLE statement is as follows:

```
CREATE TABLE schema.tablename
  (CUST_ID          NUMBER(5)          CONSTRAINT nn_hcis01_custid NOT NULL
  CUST_NAME         VARCHAR2(20)       CONSTRAINT nn_hcis01_custname NOT NULL
  CUST_PHONE        VARCHAR2(10))
```

```
TABLESPACE inventory
STORAGE (INITIAL 6144
        NEXT 6144
        MINEXTENTS 1
        MAXEXTENTS 5
        PCTINCREASE 0);
```

6.0 Storage

The storage clause specifies the storage characteristics of Oracle objects such as tables and indexes. Storage parameters affect both how long it takes to access data stored in the database and how efficiently space in the database is used.

When you create a table or index, you can specify values for the storage parameters for the segments allocated to these objects. If you omit any storage parameter, Oracle uses the value of that parameter specified for the tablespace.

When you alter an index or table, you can change the values of storage parameters. The new values affect only future extent allocations.

To change the value of a STORAGE parameter, you must have the privileges necessary to use the appropriate CREATE or ALTER statement.

Development and test databases are usually much smaller than a production database and storage requirements should be defined accordingly.

6.1 Object Usage

Construct the STORAGE clause of a table or an index as follows:

- INITIAL = Oracle blocks needed to store a fully populated table or index in production;
- NEXT = Oracle blocks needed to store the incremental growth at some regular frequency, such as monthly or quarterly;
- PCTINCREASE = 0.

6.2 Syntax

Parameters:

- INITIAL specifies the size in bytes of the object's first extent. Oracle allocates space for this extent when you create the schema object. The default value is the size of 5 data blocks. The minimum value is the size of 2 data blocks. **Note:** You cannot specify INITIAL in an ALTER statement.
- NEXT specifies the size in bytes of the next extent to be allocated to the object. The default value is the size of 5 data blocks. The minimum value is the size of 1 data block. If you change the value of the NEXT parameter (that is, if you specify it in an ALTER statement), the next allocated extent will have

the specified size, regardless of the size of the most recently allocated extent and the value of the PCTINCREASE parameter.

- PCTINCREASE specifies the percent by which the third and subsequent extents grow over the preceding extent. The default value is 50, meaning that each subsequent extent is 50% larger than the preceding extent. The minimum value is 0, meaning all extents after the first are the same size. Oracle rounds the calculated size of each new extent up to the next multiple of the data block size. If you change the value of the PCTINCREASE parameter (that is, if you specify it in an ALTER statement), Oracle calculates the size of the next extent using this new value and the size of the most recently allocated extent. If you wish to keep all extents the same size, you can set the value of PCTINCREASE to 0.
- MINEXTENTS specifies the total number of extents to allocate when the object is created. This parameter enables you to allocate a large amount of space when you create an object, even if the space available is not contiguous. The default and minimum value is 1. If the MINEXTENTS value is greater than 1, then Oracle calculates the size of subsequent extents based on the values of the INITIAL, NEXT, and PCTINCREASE parameters. You cannot specify MINEXTENTS in an ALTER statement.
- MAXEXTENTS specifies the total number of extents, including the first, that Oracle can allocate for the object. The minimum value is 1. The default and maximum values depend on your data block size.
- UNLIMITED specifies that extents should be allocated automatically as needed. Transactions containing inserts, updates, or deletes that continue for a long time will continue to create new extents until a disk is full.

7.0 Columns

Oracle tables are made up of columns that contain data that was loaded, inserted, or updated by some process. All columns have a corresponding datatype to indicate the format of the data contained in that column. In addition to datatype, other edits can be associated with columns in Oracle to enforce defined business rules. These edits include default values, check constraints, unique constraints, and referential integrity (foreign keys).

7.1 Column Naming Standards

See Standard Naming Conventions in the Oracle standards and *Creating Physical Names for Elements and Columns* in the Data Administration standards.

7.2 Object Usage

Columns that represent the same data but are stored on different tables must have the same name, datatype and length specification. An example of this is where tables are related referentially. If on a PROVIDER table, for example, the primary key is defined as PROV_NUM CHAR(08), then dependent tables must include PROV_NUM CHAR(08) as a foreign key column.

Columns that contain data that is determined to have the same domain must be defined using identical datatype and length specifications. For example, columns LAST_CHG_USER_ID and CASE_ADMN_USER_ID serve different business purposes, however both should be defined as CHAR (08) in DB2. This standard applies to the entire enterprise and should not be enforced solely at an application level.

All columns containing date information must be defined using the DATE data type.

Specify a datatype that most represents the data the column will contain. For numeric data, use one of the supported numeric data types taking into account the minimum and maximum value limits as well as storage requirements for each.

For character data that may exceed 20 characters in length, consider use of the VARCHAR(2) datatype. This could provide substantial savings in storage requirements. When weighing the benefits of defining a column to be variable in length, consider the average length of data that will be stored in this column. If the average length is less than 80% of the total column width, a variable length column may be appropriate.

Consider sequence of the column definitions to improve database performance. Use the following as a guideline for sequencing columns on Oracle tables.

1. Primary Key columns (for reference purposes only)
2. Frequently read columns
3. Infrequently read columns
4. Infrequently updated columns
5. Variable length columns
6. Frequently updated columns

For columns defined as DECIMAL be sure to use an odd number as the precision specification to ensure efficient storage utilization. Precision represents the entire length of the column, so in the definition DECIMAL (9,2), the precision is 9.

For columns with whole numbers SMALLINT and NUMBER shall be used.

8.0 Indexes

An index is an ordered list of all the values that reside in a column or columns of a table. An index greatly increases the efficiency of a query running against the data within the column, returning the results of the query more quickly. Oracle can use indexes to improve performance when searching for rows with specified index column values and when accessing tables in index column order.

8.1 Object Usage

When rows are initially inserted into a new table, it is generally faster to create the table, insert the rows, and then create the index. If the index is created before inserting the rows, Oracle must update the index for every row inserted.

If an index is required, it must always be created in a tablespace reserved for indexes only.

An index can contain a maximum of 32 columns. The index entry becomes the concatenation of all data values from each column. You can specify the columns in any order. The order you choose is important to how Oracle uses the index.

Do not index columns that are frequently modified. UPDATE statements that modify indexed columns and INSERT and DELETE statements that modify indexed tables take longer than if there were no index. Such SQL statements must modify data in indexes as well as data in tables. They also generate additional undo and redo information.

Do not index columns that appear only in WHERE clauses with functions or operators. A WHERE clause that uses a function (other than MIN or MAX), or an operator with an indexed column, does not make available the access path that uses the index.

When choosing whether to index a column, consider whether the performance gain for queries is worth the performance loss for INSERT, UPDATE, and DELETE statements and the use of the space required to store the index.

8.2 How to Choose Composite Indexes

A composite index contains more than one key column. Composite indexes can provide additional advantages over single-column indexes, i.e., better selectivity and additional data storage. Consider creating a composite index on columns that are frequently used together in WHERE clause conditions combined with AND operators. Also consider the guidelines associated with the general performance advantages and trade-offs of indexes.

Create the index so that the columns that are used in WHERE clauses make up a leading portion.

If some of the columns are used in WHERE clauses more frequently, be sure to create the index so that the more frequently selected columns make up a leading portion to allow the statements that use only these columns to use the index.

If all columns are used in WHERE clauses equally often, ordering these columns from most selective to least selective in the CREATE INDEX statement best improves query performance.

If all columns are used in the WHERE clauses equally often but the data is physically ordered on one of the columns, place that column first in the composite index.

8.3 Multiple Indexes Per Table

You can create unlimited indexes for a table provided that the combination of columns differs for each index. You can create more than one index using the same columns provided that you specify distinctly different combinations of the columns. For example, the following statements specify valid combinations:

```
CREATE INDEX emp_idx1 ON scott.name (job);
CREATE INDEX emp_idx2 ON scott.emp (job, ename);
```

Note: you do not need to specify the owner's name when creating indexes on your own tables.

You cannot create an index that references only one column in a table if another such index already exists.

Remember that each index increases the processing time needed to maintain the table during updates to indexed data. Thus, updating a table with a single index will take less time than if the table had five indexes.

8.4 The NOSORT Option

The NOSORT option can substantially reduce the time required to create an index. Normal index creation first sorts the rows of the table based on the index columns and then builds the index. The sort operation is often a substantial portion of the total work involved. If the rows are already physically stored in ascending order (based on the indexed column values), then the NOSORT option causes Oracle to bypass the sort phase of the process.

To use the NOSORT option, you must guarantee that the rows are physically sorted in ascending order. If your rows are not in ascending order, Oracle returns an error. You can issue another CREATE INDEX without the NOSORT option. Because of the physical data independence inherent in relational database management systems, especially Oracle, there is no way to force a physical internal order on a table. The CREATE INDEX command with the NOSORT option should be used immediately after the initial load of rows into a table.

You cannot use the NOSORT option to create a cluster index, partitioned index, or bitmap index.

The NOSORT option also reduces the amount of space required to build the index. Oracle uses temporary segments during the sort. Since a sort is not performed, the index is created with much less temporary space.

8.5 NOLOGGING

The NOLOGGING option may substantially reduce the time required to create a large index. This feature is useful after creating a large index in parallel.

Example: To quickly create an index in parallel on a table that was created using a fast parallel load (so all rows are already sorted), you might issue the statement below.

```
CREATE INDEX idx1
ON big_tab1 (akey)
NOSORT
NOLOGGING
PARALLEL (DEGREE 5);
```

8.6 Nulls

Oracle does not index table rows in which all key columns are NULL, except in the case of bitmap indexes.

Example:

```
SELECT ename
FROM emp
WHERE comm IS NULL;
```

The above query does not use an index created on the COMM column unless it is a bitmap index.

8.7 Creating Partitioned Indexes

Indexes can be local prefixed (unique or nonunique), local nonprefixed (unique, but only when the partitioning key is a subset of the index key or nonunique), or global prefixed (unique or nonunique). Oracle does not support global nonprefixed indexes. Local indexes are always partitioned. Global indexes can be nonpartitioned or partitioned.

Index partitions must be listed in order. For a global index, this means that the partition bound of the first partition listed must be less than the partition bound of the second partition listed, and the partition bound of the second partition listed must be less than the third, and so on. For a local index, you must list the partitions in the same order as the partitions of the underlying table to which they correspond.

Example: The statement below creates a global prefixed index STOCK_IDX on table STOCK with two partitions, one for each half of the alphabet. The index partition names are system generated.

```
CREATE INDEX stock_idx ON stock
   (stock_symbol, stock_series)
    GLOBAL PARTITION BY RANGE (stock_symbol)
     (PARTITION VALUES LESS THAN ('N') TABLESPACE ts3,
      PARTITION VALUES LESS THAN (MAXVALUE) TABLESPACE ts4);
```

8.8 Creating Bitmap Indexes

Bitmap indexes store the ROWIDs associated with a key value as a bitmap. Each bit in the bitmap corresponds to a possible ROWID, and if the bit is set, it means that the row with the corresponding ROWID contains the key value. The internal representation of bitmaps is best suited for applications with low levels of concurrent transactions, such as data warehousing.

Example: To create a bitmap-partitioned index on a table with four partitions, issue the statement below.

```
CREATE BITMAP INDEX partno_ix
ON lineitem(partno)
TABLESPACE ts1
LOCAL (PARTITION quarter1 TABLESPACE ts2,
PARTITION quarter2 STORAGE (INITIAL 10K NEXT 2K),
PARTITION quarter3 TABLESPACE ts2,
PARTITION quarter4);
```

9.0 Referential Constraints (Foreign Keys)

A referential constraint is a rule that defines the relationship between two tables. It is implemented by creating a foreign key on a dependent table that relates to the primary key (or unique constraint) of a parent table.

9.1 Foreign Key Naming Standards

See Standard Naming Conventions.

9.2 Object Usage

STANDARD:

- Names for all foreign key constraints must be explicitly defined using data definition language (DDL). Do not allow Oracle to generate default names.
- DDL syntax allows for foreign key constraints to be defined along side of the corresponding column definition, as a separate clause at the end of the table definition, or in an ALTER TABLE statement. Each of these methods is acceptable at CMS provided all of the required parameters noted below are included in the definition.
- When possible, attempt to limit the number of levels in a referential structure (all tables which have a relationship to one another) to three.
- Define indexes on foreign keys.

9.3 Required Parameters (DDL Syntax)

STANDARD: The parameters listed below must be included in the Oracle data definition language (DDL) when defining a foreign key constraint in an application table. Oracle default settings must not be assumed for any of these parameters. For additional information see Oracle's Online Reference Library.

Parameter	Instructions
CONSTRAINT constraint name	Specify a name for the foreign key constraint based on the *Standard Naming Conventions*.
FOREIGN KEY (column name...)	Indicate the column(s) that make up the foreign key constraint. These columns must correspond to a primary key or unique constraint in the parent table.
REFERENCES table name	Specify the name of the table to which the foreign key constraint refers. If the foreign key constraint is based on a unique constraint of a parent table, also provide the column names that make up the corresponding unique constraint.
ON DELETE delete rule	Specify the appropriate delete rule that should be applied whenever an attempt is made to delete a corresponding parent row. If you omit this clause, Oracle does not allow you to delete referenced key values in the parent table that have dependent rows in the child table. Valid values include SET NULL and CASCADE. *(**Warning:** Use of the CASCADE delete rule can result in the mass deletion of numerous rows from dependent tables when one row is deleted from the corresponding parent. No response is returned to the deleting application indicating the mass delete occurred. Therefore, strong consideration should*

	be given as to the appropriateness of implementing this rule in the physical design.)

10.0 Temporary Tables

Within Oracle, you can create a table for use only in your session or whose data lasts for the duration of the transaction you are conducting. These types of tables are known as temporary global tables or temporary tables. Unlike a permanent table, a temporary table does not allocate storage in a permanent tablespace, but uses sort space to store the data. Should that space fill up, additional extents are created within the user's temporary tablespace.

10.1 Object Usage

When you create a temporary table, you must specify whether you want the data to persist for the transaction or the entire session. The clauses that control the duration of the rows are:

- ON COMMIT DELETE ROWS—specifies that rows are only visible within the transaction;
- ON COMMIT PRESERVE ROWS—specifies that rows are visible for the entire session.

Indexes built upon temporary tables have the same store as the data that resides within them. Triggers and views can be defined upon temporary tables; however, a view built upon a join between temporary and permanent tables is not allowed. Definitions of temporary tables may be exported and imported.

10.2 Syntax

```
create global temporary table temp_emp
   (empno number,
   name varchar2(20) ,
   salary number)
on commit delete rows;
```

CREATE GLOBAL TEMPORARY TABLE AS SELECT is another method of creating a temporary table.

11.0 Views

A view in Oracle is an alternative representation of data from one or more tables or views. Views do not contain data. Actual data is stored with the underlying base table(s). Views are generally created to solve business requirements related to security or ease of data access. A view may be required to limit the columns or rows of a particular table a class of business users are permitted to see. Views are also created to simplify user access to data by resolving complicated SQL calls in the view definition.

11.1 Naming Standards for Views

See Standard Naming Conventions.

11.2 Object Usage

STANDARD:

- View definitions must reference base tables only. Do not create views which reference other views.
- Use views sparingly. Create views only when it is determined that direct access to the actual table does not adequately serve a particular business need.

11.3 Required Parameters (DDL Syntax)

STANDARD: The parameters listed below must be included in the Oracle data definition language (DDL) when defining a view. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference library.

Parameter	Instructions
CREATE VIEW view name	Specifies the view name. OR REPLACE may be used in addition to CREATE VIEW. This will recreate the view without dropping, re-creating, and regranting object privileges.
OF type_name	Explicitly creates an object view of type_name.
WITH OBJECT IDENTIFIER identifier name	Specifies the attributes of the object type that will be used as a key to identify each row in the object view.
AS select statement	Defines the contents of this view. Do not include existing views as part of the view definition.
WITH CHECK OPTION	Specifies that updates and inserts performed through the view must result in rows that the view can select.
CONSTRAINT constraint name	Used in conjunction with the 'WITH CHECK OPTION'. Assigns the name of the constraint.
WITH READ ONLY	Specifies that no deletes, inserts, or updates can be performed through the view.

12.0 Materialized Views

Materialized views are schema objects that can be used to summarize, precompute, replicate, and distribute data. They are suitable in various computing environments such as data warehousing, decision support, and distributed or mobile computing.

The tables in the query are called master tables or detail tables. The databases containing the master tables are called the master databases.

12.1 Naming Standards for Materialized Views

See Standard Naming Conventions.

12.2 Object Usage

STANDARD:

- Either dimensions should be denormalized (each dimension contained in one table), or the joins between tables in a normalized or partially normalized dimension should guarantee that each child-side row joins with one and only one parent-side row.
- For all dimensions, each child key value must uniquely identify its parent key value.
- Use the VALIDATE_DIMENSION procedure of the DBMS_OLAP package to verify hierarchical integrity in a denormalized dimension.
- Fact tables and dimension tables should similarly guarantee that each fact table row joins with one and only one dimension table row.

12.3 Required Parameters (DDL Syntax)

STANDARD: The parameters listed below must be included in the Oracle data definition language (DDL) when defining a view. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference library.

Parameter	Instructions
CREATE MATERIALIZED VIEW View name	Specifies the view name. SNAPSHOT may be used in place of MATERIALIZED VIEW. Limit the view name to 19 characters.
INITIAL size parameter	Specifies in bytes the size of the objects first extent. Use K or M. You cannot specify INITIAL in an ALTER statement.
NEXT size parameter	Specifies in bytes the size of the objects next extent. Use K or M. For rollback segments, use the same INITIAL and NEXT values.
MINEXTENTS	Specifies the total number of extents to allocate when the object is created. If the MINEXTENTS value is greater than 1, then Oracle calculates the size of the subsequent extents based on the values of the INITIAL, NEXT, and PCTINCREASE parameters.
MAXEXTENTS	Specifies the total number of extents, including the

	first that Oracle can allocate for the object.
PCTINCREASE	Specifies the percent by which the third and subsequent extents will grow over the preceding extent. Set PCTINCREASE to 0. This will prevent exponential growth during extent allocation.
TABLESPACE tablespace name	Specify the tablespace name for the MATERIALIZED VIEW. Objects in the SYSTEM TABLESPACE will be dropped without warning.
USING INDEX	Specifies parameters for the indexes Oracle creates to maintain the materialized view.

13.0 Synonyms

A synonym is an alias for any table, view, snapshot, sequence, procedure, function, or package. Because a synonym is simply an alias, it requires no storage other then its definition in the data dictionary. Synonyms are used for security and convenience. They mask the name and owner of an object, provide location transparency for remote objects of a distributed database, and simplify SQL statements for database users.

13.1 Object Usage

Synonyms may be public or private synonyms. A public synonym is owned by the special user group named PUBLIC and every user in a database can access it. A private synonym is in the schema of a specific user who has control over its availability to others.

13.2 Syntax

Create a synonym using the SQL command CREATE SYNONYM:

CREATE PUBLIC SYNONYM public_emp FOR schema.emp;

Drop a synonym that is no longer required using the SQL command DROP SYNONYM. To drop a private synonym, omit the PUBLIC keyword; to drop a public synonym, include the PUBLIC keyword:

DROP SYNONYM emp;

The following statement drops the public synonym named PUBLIC_EMP:

DROP PUBLIC SYNONYM public_emp;

When you drop a synonym, its definition is removed from the data dictionary. All objects that reference a dropped synonym remain; however, they become invalid (not usable).

14.0 See Also

See also the design standards for Stored Procedures.

Standard Naming Conventions

1.0 Object and Dataset Names

This section discusses the standard naming conventions for Oracle objects and datasets. These conventions were designed to meet the following objectives:

- Guarantee uniqueness of Oracle object names within an Oracle database;
- Provide a uniform naming structure for Oracle objects of similar types;
- Simplify physical database design decisions regarding naming strategies;
- Provide ability to visually group Oracle objects by the application and/or subject area which the objects were designed to support.

1.1 Usage

The following naming standards apply to any Oracle object (database, tablespace, table, view, PL/SQL routine, etc.) used to hold or maintain user data, as well as to external user objects (files, directories, script names, etc.) that will be used in conjunction with Oracle as part of standard application development and system operation procedures.

1.2 Standard Naming Format

All Oracle objects will be named according to the standard formats listed in the table below. All object names must begin with an alphabetic character.

Oracle Object Type	Required Attributes of Name	Example
Instance	• Up to 8 character name; • Begins with the application identifier; • Ends with **dev** if a development database, or **prd** if a production database; • Must be unique within server environment.	Admdev, admprd
Database	• Same as the instance name and follows the instance naming conventions; • Must be unique within the instance.	Admdev, admprd
Tablespace	• Up to 25 character name; • Begins with the application identifier and ends with	adm_data, adm_indx

	○ **_data** if the tablespace holds data; ○ **_indx** if the tablespace holds index data; • Must be unique within database.	
Table	• 30 character name (maximum); • Must be descriptive as it applies to the application; • Must end with **_tbl** identifier; • Cannot contain reserved words or special characters.	Prvdr_tbl
Index	• 30 character name (maximum); • Must be unique within database; • Must be the table name, suffixed by the type of index: ○ A primary key index should be suffixed with a **_pk##** identifier (## = sequence number); ○ A foreign key index should be suffixed with a **_fk##** identifier (## = sequence number); ○ A unique key index should be suffixed with a **_uk##** identifier (## = sequence number); ○ An alternate key index should be suffixed with a **_ak##** identifier (## = sequence number).	Prvdr_pk01 Prvdr_fk01, Prvdr_fk02 Prvdr_cd_uk01 Prvdr_cd_ak01
View	• 30 character name (maximum); • Must end with **_vw** identifier.	State_table_vw
Synonym	• 30 character (maximum).	Hsc01t
Column	• 30 character (maximum); • Unique within the corresponding table or view; • Should be derived from the business name identified during the business/data analysis process; • Each word within the column	Provider_id

	name must be separated by an underscore (_); • Column name may not contain reserved words or special characters; • All abbreviated words must be obtained from or recorded in the list of CMS standard abbreviations. For more information, see *Creating Physical Names* in the Data Administration Standards.	
Column Constraints	• Column check constraints must be prefixed by **ck**_, followed by an application identifier and underscore, followed by the column name.	Ck_adm_provider_id
Primary Key	• 30 character name (maximum); • Primary key name must be the table name suffixed by _**pk**; • Primary key name must be unique within the corresponding table definition.	Provider_pk
Foreign Key	• 30 character name (maximum); • Foreign key name must be the table name suffixed by _**fk**; • Foreign key name must be unique within the corresponding table definition.	Provider_fk01
Procedure, Function, Package	• 30 character name (maximum), • Prefixed by application identifier, followed by a description; • Suffixed by _**proc** (procedure), _**func** (function), or _**pkg** (package).	Hcis_get_cust_id_proc Hcis_get_cust_id_func Hcis_get_cust_id_pkg
Schema	• 8 character name (maximum), usually the application identifier	hcis

2.0 File Names

This section specifies the naming convention for standard library names to be used in Oracle application systems.

2.1 File Naming Convention

The file naming convention for database files is the tablespace name, followed by a two digit sequential number, depending upon the number of datafiles assigned to that tablespace, followed by '.dbf', which stands for "database file." The database file should be contained within a parent directory that is the application ID.

Individual scripts, PL/SQL routines, etc., must be uniquely named for the server to avoid multiple copies of the same script residing on the same server.

2.2 Sample File Name

A database file which is within the HCIS system and is being used as the second file for the REF tablespace would be named as follows:
/db501/hcis/ref02.dbf

3.0 Utility File Names and Script Names

Oracle tables are often loaded with files utilizing SQL LOADER. These files should uniquely identify the particular table that is being loaded with the given file. The target table should be identifiable by examining the file name.

Utility script names should readily identify the purpose and nature of the script. Utility scripts should also be fully documented by utilizing comment blocks within the script itself.

Oracle Standards: Packages

1.0 Overview

Packages encapsulate related procedures, functions, associated cursors and variables together as a unit in the database.

You create a package in two parts: the specification and the body. A package's specification declares all public constructs of the package, and the body defines all constructs (public and private) of the package. This separation of the two parts provides the following advantages:

- The developer has more flexibility in the development cycle. You can create specifications and reference public procedures without actually creating the package body.
- You can alter procedure bodies contained within the package body separately from their publicly declared specifications in the package specification. As long as the procedure specification does not change, objects that reference the altered procedures of the package are never marked invalid. That is, they are never marked as needing recompilation.

2.0 Naming Standards

See Standard Naming Conventions.

3.0 Object Usage

STANDARD: Packages are used to define related procedures, variables, and cursors and are often implemented to provide advantages in the areas discussed below.

- Encapsulation of related procedures and variables: Packages allow you to encapsulate or group stored procedures, variables, datatypes, and so forth in a single named, stored unit in the database. This strategy provides better organization during the development process. Encapsulation of procedural constructs in a package also makes privilege management easier. Granting the privilege to use a package makes all constructs of the package accessible to the grantee.

- Declaration of public and private procedures, variables, constants, and cursors: The methods of package definition allow you to specify which variables, cursors, and procedures are public and which are private.

    - Public items are directly accessible to the user of a package.
    - Private items are hidden from the user of a package.

    For example, a package might contain ten procedures. You can define the package so that only three procedures are public and therefore available for execution by a user of the package; the remainder are private and can only be accessed by the procedures within the package.

    Do not confuse public and private package variables with grants to PUBLIC.

- Better Performance: An entire package is loaded into memory when a procedure within the package is called for the first time. This load is completed in one operation, as opposed to the separate loads required for standalone procedures. Therefore, when calls to related packaged procedures occur, no disk I/O is necessary to execute the compiled code already in memory.

- A package body can be replaced and recompiled without affecting the specification. As a result, schema objects that reference a package's constructs (always via the specification) need not be recompiled unless the package specification is also replaced. By using packages, unnecessary recompilations can be minimized, resulting in less impact on overall database performance.

4.0 Required Parameters (DDL Syntax) - Package Specification

STANDARD: The parameters listed below must be included in the Oracle data

definition language (DDL) when defining a Package. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference manuals.

Parameter	Instructions
CREATE/OR REPLACE	Specifies the creation of a new package. 'OR REPLACE' may be used to recreate an existing package. Use this clause to change the specification of an existing package without dropping, re-creating, and regranting object privileges previously granted on the package. If you change a package specification, Oracle recompiles it.
PACKAGE schema.package_name	Identifies the schema owner and package name. Omitting the 'schema' will create a package_name with a schema the same as the user creating the package.
AS/IS	Identifies the beginning of the package specification.
PL/SQL_package_spec	Specify the package specification, which can contain type definitions, cursor declarations, variable declarations, constant declarations, exception declarations, PL/SQL subprogram specifications, and call specifications (declarations of a C or Java routine expressed in PL/SQL).
END package_name	Closes the package specification.

Oracle Standards: Stored Procedures/Functions

1.0 Overview

A procedure or function is a schema object that consists of a set of SQL statements and other PL/SQL constructs, grouped together, stored in the database, and executed as a unit to solve a specific problem or perform a set of related tasks. Procedures and functions permit the caller to provide parameters that can be input only, output only, or input and output values.

Procedures and functions are identical except that functions always return a single value to the caller, while procedures do not.

A procedure has two parts: the specification and the body.

- The specification (spec for short) begins with the keyword PROCEDURE and ends with the procedure name or a parameter list. Parameter declarations are optional. Procedures that take no parameters are written without parentheses.

- The procedure body begins with the keyword IS (or AS) and ends with the keyword END followed by an optional procedure name. The body has three

parts: an optional declarative part, an executable part, and an optional exception-handling part.

- The declarative part contains declarations of types, cursors, constants, variables, exceptions, and subprograms. These items are local and cease to exist when you exit the procedure.
- The executable part contains statements that assign values, control execution, and manipulate Oracle data.
- The exception-handling part contains handlers that deal with exceptions raised during execution.

2.0 Naming Standards

See Standard Naming Conventions.

3.0 Object Usage

STANDARD:

- Define procedures to complete a single, focused task. Do not define long procedures with several distinct subtasks, because subtasks common to many procedures might be duplicated unnecessarily in the code of several procedures.
- Do not define procedures that duplicate the functionality already provided by other features of Oracle. For example, do not define procedures to enforce simple data integrity rules that you could easily enforce using declarative integrity constraints.
- Use Packages to encapsulate Stored Procedures.
- Stored Procedures must be commented and must include the following for the original version and each revision:
  - Author's initials
  - Date of the change
  - Reason for the change (for the original procedure, the reason is 'Original')
- All Stored Procedures must contain an 'EXCEPTION' handling section.
- Reserved words must be capitalized, all other words lowercase.
- Each Stored Procedure must have a descriptive name.

4.0 Required Parameters (DDL Syntax) - Stored Procedure Specification

STANDARD: The parameters listed below must be included in the Oracle data

definition language (DDL) when defining a a Stored Procedure Specification. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference manuals.

Parameter	Instructions
CREATE/REPLACE	Optional keywords used to create a stand-alone procedure.
PROCEDURE procedure-name	Identifies the name of the procedure.
Parameter declaration	List of parameters that are defined to pass information into and send information out of the procedure, back into the calling program.
IN, OUT, IN OUT	Part of the parameter declaration that defines the behavior of the parameter. 'IN' allows you to pass values into the subprogram; 'OUT' allows you to return values from the subprogram; 'IN OUT' allows you to pass initial values into the subprogram and return updated values. 'IN' is the default if not specified.
datatype	Part of the parameter declaration that defines the type of the parameter (NUMBER, VARCHAR2, REAL, etc.).
RETURN datatype	Used for functions, it is the datatype returned by the function—e.g., NUMBER, BOOLEAN.

5.0 Required Parameters (DDL Syntax) - Stored Procedure Body

STANDARD: The parameters listed below must be included in the Oracle data definition language (DDL) when defining a Stored Procedure Body. Oracle default settings must not be assumed for any of these parameters.

For additional information see Oracle's Online Reference manuals.

Parameter	Instructions
IS/AS	Keywords 'IS' or 'AS' identifying the beginning of the procedure body.
Declaration statements	Declarations of local identifiers for the procedure.
BEGIN	Keyword identifying the beginning of the procedure's executable statements.
EXCEPTION	Keyword identifying the beginning of exception handling.
END procedure name	Keyword identifying the end of the procedure body.

Oracle Security Standards

1.0 Overview

CMS has some basic Oracle security requirements for all new applications coming into the CMS environment. Any exceptions to these base policies must be approved by the System Security Group (OIS/SSG) and the Division of Data Management and Support Services (OIS/EDG/DDMSS). Other questions related to Oracle security should be directed to DDMSS.

2.0 Oracle Security Requirements

Requirement	Comments
All Users for the application must be authenticated to the Database.	The application interface is required to capture the user logon and password to authenticate that user to the database when accessing the Oracle database on the users' behalf. The user must have a valid Oracle user ID and password on the database, and any database activity must be performed under the User's ID for audit purposes.
Application User Interface (UI) must support Oracle database ID expiration and password change dialog.	Users of the application must be able to change their Oracle database password through the application's UI.
Master Database ID's are not permitted.	Individual database logins are required for each user.
Shared Database ID's are not permitted.	Individual database logins are required for each user.
All Database users must be assigned to specific database roles.	Users cannot be assigned individual object privileges.
No application users will be assigned database administrator roles.	Reserved for DBA staff.
Production Control Database ID's are permitted.	Must be cleared with DDMSS.
Logon level auditing must be implemented on all databases.	See DDMSS for assistance.
Oracle logon ID's will be assigned by the CMS RACF administrator.	